SYSTEM AND METHOD FOR SELECTIVELY SUGGESTING GOODS OR SERVICES

CROSS-REFERENCE TO RELATED APPLICATION

This application is related to co-pending U.S. provisional patent application entitled "Intelligent Rule-Based Promotion Module," filed on June 24, 2002, and accorded serial number 60/390,953, which is entirely incorporated herein by reference.

FIELD OF THE INVENTION

The present invention is generally related to a system and method for selectively suggesting goods or services.

BACKGROUND OF THE INVENTION

In most retail-ordering scenarios, promotions and up-selling are key tools used by sales personnel to increase price of an order, or to familiarize customers with new products. Other customer relationship tools used to increase sales include coupons and advertisements offering discounts to attract certain demographic groups, such as people over sixty-five, families with children, and frequent buyers.

Typical automated self-service terminals, also referred to herein as automated self-service systems, are incapable of replacing human sales personnel and advertisements in functions such as promotion and up-selling activities. Typical automated self-service systems also lack the ability to discriminate conditions and carefully target promotions or up-sells based on circumstances surrounding a sale. When deciding if up-sell opportunity exists, these automated self-service systems fail to consider elements, such as: the weather; the time of the day or year; the customer's age or sex; the customer's martial or parental status; the content of the current order; the customer's or his/her family's previous buying patterns; demographic information;

market research; previous responses of this customer to up-sell attempts; whether the customer is near home or traveling; whether the customer has ever tried a particular product; and results of an interactive game or random lottery. Hence, typical automated self-service systems either miss opportunities to maximize revenue, or annoy customers by asking irrelevant questions.

Thus, a heretofore unaddressed need exists in the industry to provide an automated self-service system that is responsive to customer preferences and overcomes the aforementioned deficiencies and inadequacies.

SUMMARY OF INVENTION

Embodiments of the present invention provide a system and method for selectively suggesting goods or services. Briefly described, in architecture, one embodiment of the system, among others, can be implemented as follows. The system contains a memory and a processor, configured by the memory to perform the steps of: receiving data from a customer, wherein the data specifies goods or services that are selected by the customer; and determining a suggestion to make to the customer via use of the data, the suggestion being based upon information associated with the customer.

The present invention can also be viewed as providing methods for selectively suggesting goods or services. In this regard, one embodiment of such a method, among others, can be broadly summarized by the following steps: receiving data from a customer, wherein the data specifies goods or services that are selected by the customer; and determining a suggestion to make to the customer via use of the data, the suggestion being based upon information associated with the customer.

Other systems, methods, features and advantages of the present invention will be or become apparent to one with skill in the art upon examination of the following drawings and detailed description. It is intended that all such additional systems, methods, features, and advantages be included within this description, be within the scope of the present invention, and be protected by the accompanying claims.

BRIEF DESCRIPTION OF THE DRAWINGS

Many aspects of the invention can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the present invention. Moreover, in the drawings, like reference numerals designate corresponding parts throughout the several views.

FIG. 1 is a block diagram illustrating a general purpose computer that can implement the present automated self-service system.

FIGS. 2-3 are screen illustrations provided in accordance within the present invention.

FIG. 4 is a logical diagram that provides an overview of interaction with the automated self-service system of FIG. 1.

FIG. 5 is a screen view that could be provided to a customer placing an order.

FIG. 6 is a flow chart illustrating an exemplary method of cascading rules.

FIG. 7 is a flow chart illustrating an second exemplary method of cascading rules.

FIG. 8 is a flow chart illustrating an third exemplary method of cascading rules.

FIG. 7 is another flow diagram consistent with another exemplary logic model of the present invention.

FIG. 8 is another flow diagram consistent with another exemplary logic model of the instant invention.

DETAILED DESCRIPTION

The present invention is designed for businesses that provide goods and/or services to customers, where the businesses would benefit from the ability to customize sales suggestions. Specifically, the invention provides a system & method for selectively suggesting goods & services.

The automated self-service system of the invention can be implemented in software (*e.g.*, firmware), hardware, or a combination thereof. The automated self-service system is preferably implemented in software, as an executable program, and is executed by a special or general purpose digital computer, such as a personal computer (PC; IBM-compatible, Apple-compatible, or otherwise), workstation, minicomputer, or mainframe computer. An example of a general purpose computer that can implement the automated self-service system of the present invention is shown in the block diagram of FIG. 1. In FIG. 1, the software that defines functionality performed by the automated self-service system is denoted by reference numeral 100.

Generally, in terms of hardware architecture, as shown in FIG. 1, the computer 10 includes a processor 12, memory 14, and one or more input and/or output (I/O) devices 16 (or peripherals) that are communicatively coupled via a local interface 18. The local interface 18 can be, for example but not limited to, one or more buses or other wired or wireless connections, as is known in the art. The local interface 18 may have additional

elements, which are omitted for simplicity, such as controllers, buffers (caches), drivers, repeaters, and receivers, to enable communications. Further, the local interface may include address, control, and/or data connections to enable appropriate communications among the aforementioned components. It should be noted that the computer 10 may also have a storage device 20 therein. The storage device 20 may be any non-volatile memory element (*e.g.*, ROM, hard drive, tape, CDROM, etc.).

The processor 12 is a hardware device for executing the software 100, particularly that stored in memory 14. The processor 12 can be any custom made or commercially available processor, a central processing unit (CPU), an auxiliary processor among several processors associated with the computer 10, a semiconductor based microprocessor (in the form of a microchip or chip set), a macroprocessor, or generally any device for executing software instructions. Examples of suitable commercially available microprocessors are as follows: a PA-RISC series microprocessor from Hewlett-Packard Company, an 80x86 or Pentium series microprocessor from Intel Corporation, a PowerPC microprocessor from IBM, a Sparc microprocessor from Sun Microsystems, Inc, or a 68 automated self-service series microprocessor from Motorola Corporation.

The memory 14 can include any one or combination of volatile memory elements (*e.g.*, random access memory (RAM, such as DRAM, SRAM, SDRAM, *etc.*)) and nonvolatile memory elements. Moreover, the memory 14 may incorporate electronic, magnetic, optical, and/or other types of storage media. Note that the memory 14 can have a distributed architecture, where various components are situated remote from one another, but can be accessed by the processor 12.

The software 100 located in the memory 14 may include one or more separate programs, each of which comprises an ordered listing of executable instructions for implementing logical functions. In the example of FIG. 1, as mentioned above, the software 100 includes functionality performed by the automated self-service system in accordance with the present invention and a suitable operating system (O/S) 22. A nonexhaustive list of examples of suitable commercially available operating systems 22 is as follows: (a) a Windows operating system available from Microsoft Corporation; (b) a Netware operating system available from Novell, Inc.; (c) a Macintosh operating system available from Apple Computer, Inc.; (e) a UNIX operating system, which is available for purchase from many vendors, such as the Hewlett-Packard Company, Sun Microsystems, Inc., and AT&T Corporation; (d) a LINUX operating system, which is freeware that is readily available on the Internet; (e) a run time Vxworks operating system from WindRiver Systems, Inc.; or (f) an appliance-based operating system, such as that implemented in handheld computers or personal data assistants (PDAs) (*e.g.*, PalmOS available from Palm Computing, Inc., and Windows CE available from Microsoft Corporation). The operating system 22 essentially controls the execution of other computer programs, such as the automated self-service system software 100, and provides scheduling, input-output control, file and data management, memory management, and communication control and related services.

The automated self-service system software 100 is a source program, executable program (object code), script, or any other entity comprising a set of instructions to be performed. When a source program, then the program needs to be translated via a compiler, assembler, interpreter, or the like, which may or may not be included within the

insert ":" NimesA.1001 ~~Efast 02.02~~

memory 14, so as to operate properly in connection with the O/S 22. Furthermore, the automated self-service system software 100 can be written as (a) an object oriented programming language, which has classes of data and methods, or (b) a procedure programming language, which has routines, subroutines, and/or functions, for example but not limited to, C, C+ +, Pascal, Basic, Fortran, Cobol, Perl, Java, and Ada.

The I/O devices 16 may include input devices, for example but not limited to, a keyboard, mouse, scanner, microphone, touchscreens, etc. Furthermore, the I/O devices 16 may also include output devices, for example but not limited to, a printer, display, *etc.* Finally, the I/O devices 16 may further include devices that communicate both inputs and outputs, for instance but not limited to, a modulator/demodulator (modem; for accessing another device, system, or network), a radio frequency (RF) or other transceiver, a telephonic interface, a bridge, a router, *etc.* In accordance with present invention, at least one of the I/O devices 16 is a display, such as a computer screen.

If the computer 10 is a PC, workstation, or the like, the software in the memory 14 may further include a basic input output system (BIOS) (omitted for simplicity). The BIOS is a set of essential software routines that initialize and test hardware at startup, start the O/S 22, and support the transfer of data among the hardware devices. The BIOS is stored in ROM so that the BIOS can be executed when the computer 10 is activated.

When the computer 10 is in operation, the processor 12 is configured to execute the self-service system software 100 stored within the memory 14, to communicate data to and from the memory 14, and to generally control operations of the computer 10 pursuant to the software 100. The automated self-service system 100 and the O/S 22, in

whole or in part, but typically the latter, are read by the processor 12, perhaps buffered within the processor 12, and then executed.

When the automated self-service system is implemented in software 100, as is shown in FIG. 1, it should be noted that the automated self-service system software 100 can be stored on any computer readable medium for use by or in connection with any computer related system or method. In the context of this document, a computer readable medium is an electronic, magnetic, optical, or other physical device or means that can contain or store a computer program for use by or in connection with a computer related system or method. The automated self-service system software 100 can be embodied in any computer-readable medium for use by or in connection with an instruction execution system, apparatus, or device, such as a computer-based system, processor-containing system, or other system that can fetch the instructions from the instruction execution system, apparatus, or device and execute the instructions. In the context of this document, a "computer-readable medium" can be any means that can store, communicate, propagate, or transport the program for use by or in connection with the instruction execution system, apparatus, or device. The computer readable medium can be, for example but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or propagation medium. More specific examples (a nonexhaustive list) of the computer-readable medium would include the following: an electrical connection (electronic) having one or more wires, a portable computer diskette (magnetic), a random access memory (RAM) (electronic), a read-only memory (ROM) (electronic), an erasable programmable read-only memory (EPROM, EEPROM, or Flash memory) (electronic), an optical fiber (optical), and a portable

compact disc read-only memory (CDROM) (optical). Note that the computer-readable medium could even be paper or another suitable medium upon which the program is printed, as the program can be electronically captured, via for instance optical scanning of the paper or other medium, then compiled, interpreted or otherwise processed in a suitable manner if necessary, and then stored in a computer memory.

In an alternative embodiment, where the automated self-service system is implemented in hardware, the automated self-service system can be implemented with any or a combination of the following technologies, which are each well known in the art: a discrete logic circuit(s) having logic gates for implementing logic functions upon data signals, an application specific integrated circuit (ASIC) having appropriate combinational logic gates, a programmable gate array(s) (PGA), a field programmable gate array (FPGA), *etc.*

For the purposes of illustration, a software implementation of the invention will be described with reference to the quick serve food industry. However, this example in no way should be considered limiting.

In accordance with a first exemplary embodiment of the present invention, FIGS. 2 and 3 are examples of a series of menus/screens used on a self-service terminal to allow a customer to order a meal at a quick serve restaurant. It should be noted that the menu views may be viewed inside or out of the quick serve restaurant. In addition, it should be noted that the menu views are each defined by the automated self-service system software 100.

FIG. 2(A) is a screen providing a menu that allows the customer to select a food item on a touch screen panel. Using this menu, the user can easily select to see either, a

full menu, or possible combinations. FIG. 2(B) is a screen that provides possible combination meals available to the user if he or she chooses that respective option.

FIG. 2(C) – 3(B) are screens that provide options to allow a user to optionally customize the meal to the customer's personal tastes. These menus allow a user to customize their meal orders by adding different toppings, such as, but not limited to, ketchup, mustard or pickles, or options, such as, but not limited to beverage size. FIG. 3(C) is a screen that summarizes a placed order. FIG. 3(D) shows the main selection page along with a summary window in the bottom right hand corner, also referred to herein as an on-screen receipt. This on-screen receipt preferably displays the order along with the subtotal, tax and total as its being composed. This allows for easy editing and removal of items at any time by the customer. The screen of FIG. 3(D) also shows the checkout screen, which accepts different forms of payment.

FIG. 5 is a screen that could be provided to a customer placing an order, during or at the end of placing an order, to up-sell items to the customer. It should be noted that different queries may be provided to the customer during or after placing the order.

FIG. 4 is a logical diagram that provides an overview of interaction with the automated self-service system in accordance with the first exemplary embodiment of the invention to control types of up-sell items offered to the customer. If the system utilizes user identity information to formulate types of up-sell items, the user inputs into a user interaction mechanism 104, a sort identity means. Such identity means could be a rewards card, credit card, identification number, or any other means. The user then inputs the order through the user interaction mechanism 104, such as a touch screen, a computer mouse, or a different I/O device 16 (FIG. 1). The user interaction mechanism

104 communicates with an intelligent promotion mechanism/module (IPM) 102, which constructs a current order database 106 that may be provided within the storage device 20 (FIG. 1). The customer can customize an item or service by, for example, choosing sizes, options, or toppings, and these selections further update data stored within the current order database 106.

The IPM 102 then uses the current order database 106 to determine whether to offer a suggestion to the user. Different methods may be used by the IPM 102 to determine whether to offer a suggestion. One possible set of criteria includes a user profile database 110 containing attributes about a system's user (*e.g.*, age, sex, name, *etc.*), a restaurant profile database 108 containing attributes of a restaurant (*e.g.*, location), or an arbitrary set of flags and counters. It should be noted that the user profile database 110, restaurant profile database 108, and arbitrary set of flags and counters may be provided within the storage device 20 (FIG. 1).

The system may generate a message to be displayed on a screen for an operator to communicate a suggestion to the customer, on a screen for the customer to read, as shown is FIG. 5, or may be audibly communicated to the customer over a speaker. Furthermore, the system allows the customer to decide whether he/she wishes to accept or reject the suggestion.

The IPM 102 may be invoked under two circumstances: 1) after an item is chosen and optionally customized; or 2) at the end of the order, when, for example, "Checkout" is pressed.

Each of these invocations of the IPM 102 is accompanied by an invocation of a context specific rule set 114. The invoked rule set 114 is either associated with a type of

item being ordered or a "Checkout" rule set, and acts as an ordered set of rules that are sequentially executed by the IPM 102. Each rule within a rule set 114 contains two parts: a predicate (condition) that describes situations under which the rule should be activated; and a series of actions to be invoked when the rule is activated. An example of how the rule may be notated is:

Predicate ➔ Action1; Action2; … ;ActionN

(where ➔ is read as "implies" or "indicates").

An intuitive way to follow the meaning of a rule set is to read it as:

"situation" "indicates" "response"

Predicate ➔ Actions

A predicate is a boolean expression made up of boolean terms connected by "and," "or," or "not" operators. These expressions can also contain parentheses, which indicate the order of operations. A non-boolean term can be used to construct a boolean term by comparing it to a reference value (*e.g.*, nb_term > 4, nb_term == "abc", *etc.*). Comparators supported include, at least, "less than", "greater than", "less than or equal to", "greater than or equal to", "equal to", and "not equal to." Processing of a given rule is straightforward: if predicate is TRUE then each action in the action list is executed. Any rule whose predicate evaluates to FALSE is skipped, *i.e.*, the actions are not executed. Appendix B contains an exemplary set of rule terms.

Illustrative results and their corresponding rules will now be described with reference to the rule terms in Appendix B. First, a desired result might be to ask 50% of the users who did not order a beverage before 10:30 AM whether they would like a coffee with their order. The corresponding rule is:

Nimesh.1001
~~Ffast 02.02~~

Probability(.5) and not ItemExists("Beverage") and OperTime() < 10.5 ➔

SolicitAddon("Would you like a coffee with your order?", "Coffee");

A second desired result could be to ask 80% of males under 60 and anonymous users who ordered a small combo if they would like to upgrade to a large combo. The corresponding rule is:

Probability(.8) and ((IsMale() and UserAge() < 60) or IsAnonymous()) and

ItemExists("Combo", "Small") ➔ SolicitUpgrade("Would you like to change our

small combo to a large?", "Small", "Large");

A third example of a possibly advantageous result is to provide a discount of 10% to customers over the age of 65 who have a signed up for the preferred customer program. The rule is:

UserAge() >= 65 and LoyaltyLevel() >= 2 ➔ Discount(OrderValue() * .1);

Another example of a possibly desirable result is to ask users who ordered small fries if they want to upgrade to large unless we've already solicited them two or more times. The rule would be:

ItemExists("French Fries", "Small") and SolicitCount <= 2 ➔

SolicitUpgrade("Wouldn't you rather have a large French Fries?", "French Fries",

"Small", "Large");

In addition, a profitable result could be if the outside temperature is over 85 degrees and the user is a coffee drinker and did not order a beverage, to offer the customer a new 36 oz iced coffee. The rule to produce this result is:

Mesa.1001
~~Ffast 02.02~~

CurrentTemp() > 85 and UserFlag("CoffeeDrinker") and not

ItemExists("Beverage") → SolicitAddon("Would you like to try our new 36 oz.

Iced Coffee?", "Iced Coffee");

An exemplary method of cascading the rules will now be described with reference to a flow chart provided by FIG. 6. The following described flow charts shows the architecture, functionality, and operation of a possible implementation of the automated self-service system software 100 (FIG. 1). In this regard, each block represents a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that in some alternative implementations, the functions noted in the blocks may occur out of the order noted in the flow charts. For example, two blocks shown in succession in may in fact be executed substantially concurrently or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved, as will be further clarified hereinbelow.

Referring back to FIG. 6, if a known coffee drinker does not order a beverage, a desired result would be to ask if they would like coffee with their meal. The corresponding rules first require that once a customer orders coffee, then the customer is designated as a coffee drinker in the system:

ItemExists("Coffee") → SetUserFlag("CoffeeDrinker"); \\ (block 301, block 302)

Then, if a customer is a coffee drinker and did not order a beverage on subsequent visits, the system asks if he/she want coffee:

UserFlag("CoffeeDrinker") and not ItemExists("Beverage") → \\ (block 301, block 305) SolicitAddon("Would you like a coffee with your order?", "Coffee"); \\ (block 304)

Describing FIG. 6 in more detail, a customer order is processed (block 300). In this example the rules are utilized to determine if coffee is included in the customer order

(block 301). It should be noted that other questions may be provided instead. If coffee is included in the customer order, the fact that the customer ordered coffee is included in the preferences of the customer (block 302). The rules are then returned to, thereby providing for further evaluation and processing of customer actions (block 303). If, however, coffee is not included, a determination is made as to whether the customer has ordered coffee before (block 305). If the customer has not ordered coffee before, the rules are returned to, thereby providing for further evaluation and processing of customer actions (block 303). If, instead, the customer has ordered coffee before, the customer is automatically asked if he/she wants coffee (block 304). A return is then made to the rules (block 303).

Another exemplary result, illustrated by the flow chart of FIG. 7, is to ask customers who did not order a beverage with their meal before 10:30 AM if they would like a coffee with their order. If they decline repetitively (*e.g.*, three times on different days), the system stops asking them if they would like a coffee. If they order a coffee at any point again in the future, the system restarts the counter, and asks whether they want coffee on at least 3 subsequent visits. Of course, the number of subsequent visits may be different. The rules needed to obtain the aforementioned result are: ask if it is before 10:30, if beverage has been ordered and the suggestion has not been rejected three times then we offer this customer coffee:

OperTime() > 10.5 and not ItemExists("Beverage"); // (block 402 and block 404) and UserCount("RejectCoffee") <= 3 → SolicitAddon("Would you like a coffee with your order?", "Coffee"); // (block 406 and block 410)

If the customer rejected the coffee solicitation, then the rejected coffee counter is incremented:

WasRejected("Coffee") ➔ IncrementUserCount("RejectCoffee", 1); // (block 414)

If the customer's order includes coffee, then the rejected coffee counter is set to zero.

ItemExists("Coffee") ➔ SetUserCount("RejectCoffee", 0); // (block 408)

Another exemplary result, illustrated by the flow chart of FIG. 8, could be to award a free cherry pie to a customer with an order totaling over $10.00. To accomplish this result, first if the customer did not already order or receive a cherry pie, the automated self-service system software 100 enables asking if the customer wants a free cherry pie:

OrderValue > 10.0 and not ItemExists("Cherry Pie") ➔ SolicitAddon("Would you like a free cherry pie with your order?", "cherry pie"); // (block 502, block 504, and block 506)

If there is a cherry pie in the order, discount the full price (*i.e.*, give the customer the cherry pie for free):

OrderValue() > 10.0 and ItemExists("cherry pie") ➔ ItemDiscount("cherry pie", PriceOf ("cherry pie")); // (block 502, block 504 and block 510)

Another exemplary embodiment of the system may also include one or more of the following features alone or in combination: hierarchical typing, hierarchical application of rules, solicitations ganging, global boundary controls, item selection maximum solicitations, total item selection maximum solicitations, checkout maximum solicitations, and total maximum solicitations.

It should be emphasized that the above-described embodiments of the present invention are merely possible examples of implementations, merely set forth for a clear understanding of the principles of the invention. Many variations and modifications may be made to the above-described embodiment(s) of the invention without departing substantially from the spirit and principles of the invention. All such modifications and variations are intended to be included herein within the scope of this disclosure and the present invention and protected by the following claims.

APPENDIX A: DEFINITIONS

Hierarchical Typing: If the target system contains a hierarchical categorization of item types, then "type" can be any node in the hierarchy, whether it represents a category (*e.g.*, beverages) or a specific orderable item ("iced coffee"). Any of the terms that take "type" as a parameter can use it in this hierarchical sense

Hierarchical Application of Rules: A rule set may be attached at a given point in the type hierarchy. When the customer selects an item, the rule set associated with the specific orderable item is invoked (if any) followed by the rule set of the next highest category (if any) all the way up to the root of the hierarchy. This allows rule sets to be attached to abstract items (*e.g.*, beverages) rather than replicating the logic for each type of beverage, while not precluding specific rules for specific items.

Solicitation Ganging: A number of solicitations can be presented to the user at once rather than sequentially. This minimizes interaction time and lets the customer know that there is a bounded set of questions (*e.g.*, FIG. 5). The system may be designed so the screen stays until the customer presses yes or no for each of the choices. When the

customer has answered all questions, the next screen will be shown. Pressing yes or no may cause that selection to be highlighted to show the selected answer for each question.

Global Boundary Controls: Certain controls determine the behavior of the system as a whole. These can be specified globally so that individual rules do not have to enforce boundaries.

Item Selection Maximum Solicitations: The maximum number of solicitations that will be made during an item selection. Sequential rule processing for the current rule set will be aborted once this limit is reached.

Total Item Selection Maximum Solicitations: The maximum number of solicitations that will be made during the session for all item selections. Once this limit is reached, no further rule processing will be done for any item selections for the remainder of the session.

Checkout Maximum Solicitations: The maximum number of solicitations that will be made during checkout processing. Sequential rule processing will be aborted once this limit is reached.

Total Maximum Solicitations: The maximum number of solicitations that will be made during the session. Once this limit is reached, no further rule processing will be done for any item selections or checkout processing for the remainder of the session.

APPENDIX B: EXEMPLARY RULE TERMS

Exemplary rule terms follow.

- Probability(X) – Returns TRUE on a random basis with a probability of X where $0 \leq X \leq 1$

- ItemCount(T) – Returns the count of items within the order of type = T
- ItemExists(T) – Returns TRUE if at least one item of type = T exists
- UserFlag(F) – Returns TRUE if the flag labeled F from the user's profile is set to TRUE; otherwise FALSE
- OperTime(T) – Returns the floating-point number of hours since the start of the current day (i.e. 00:00:00)

  UserAge() – Returns the age (in years) of the current user (from the user profile)
- IsMale() – Returns TRUE if the current user is Male (from profile)
- IsFemale() – Returns TRUE if the current user is Female (from profile)
- IsAnonymous() – Returns TRUE if the current user has not identified his / her self

  LoyaltyLevel() – Returns the loyalty level of the current user, zero is returned for anonymous users, provides levels of preference for different customers
- OrderValue() – Returns the size (in floating-point monetary units) of the current order (less taxes)
- OrderItems() – Returns the count of items in the current order
- MonthlyValue() – Returns the monetary value of all of the orders the user has purchased in the current month (from profile)
- MonthlyOrderCount() – Returns the number of orders the customer has placed in the current month (from profile)
- RestaurantFlag(F) – Returns TRUE if the flag labeled F from the current restaurant's profile is set to TRUE; otherwise FALSE
- UserCounter(C) – Returns the value of counter C from the user's profile

The following item selection rule terms are only applicable for rule-sets invoked during item selection (*i.e.*, not applicable for the Checkout rule set).

ItemType() – The type of the current item

ItemPrice() – The price of the current item

- HasOption(O) – Returns TRUE if the current item includes option O

The following Actions may be executed:

- Discount(N) – Reduce the order price (sub-total) by N monetary units
- ItemDiscount(T, P, N) – Reduce the price of the first N items of type T by P monetary units each. If N is omitted, then all items of type T will be discounted

ItemPrice(T, P, N) – Set the price of the first N items of type T to P monetary units, if N is omitted, then all items of type T will be repriced

SetUserCounter(C, N) – Set the value of the counter C in the user profile to N

IncrementUserCounter(C, I) – Increment the value of counter C in the user profile by I

- DecrementUserCounter(C, I) – Decrement the value of counter C in the user profile by I
- SetRestaurantCounter(C, N) – Set the value of the counter C in the restaurant profile to N
- IncrementRestaurantCounter(C, I) – Increment the value of counter C in the restaurant profile by I
- DecrementRestaurantCounter(C, I) – Decrement the value of counter C in the restaurant profile by I
- SetUserFlag(F) – Set the value of flag F in the user profile to TRUE

- ClearUserFlag(F) – Set the value of flag F in the user profile to FALSE
- SetRestaurantFlag(F) – Set the value of flag F in the restaurant profile to TRUE
- ClearRestaurantFlag(F) – Set the value of flag F in the restaurant profile to FALSE
- SolicitUpgrade(Q, T, R, A) – Present the message Q to the user and solicit yes or no answer, if the user accepts, remove the R option of an item of type T and replace it with the A option (*e.g.*, replace small with large)
- SolicitAddon(Q, T) -- Present the message Q to the user and solicit yes or no answer, if the user accepts, add an item of type T